Proceedings of the Southern Conference on Computing
The University of Southern Mississippi, Oct. 26-28, 2000

PARALLELIZATION OF FINITE ELEMENT METHOD FLOW AND TRANSPORT GROUNDWATER COMPUTATIONS

Fred Tracy*

Abstract. Finite element method flow and transport groundwater computations are very challenging because of the highly nonlinear nature of the unsaturated flow equations. It is even more difficult to produce parallel versions of the associated computer program. This paper describes the parallelization of a groundwater program called FEMWATER using the Message Passing Interface (MPI) paradigm. It also addresses some computational issues and solutions obtained that arose during this process, such as efficiency of solvers. FEMWATER does a Galerkinn finite element formulation for flow and uses an Eulerian-Lagrangian approach for transport. Various boundary conditions are allowed, as well as density dependent coupled flow and transport for modeling salt-water intrusion. Finally, an example of a flow problem will be given to demonstrate some results of this effort.

1. Introduction. FEMWATER (Lin, Richards, Talbot, Yeh, Cheng, Cheng, and Jones, 1997) is a legacy finite element program in the Groundwater Modeling System (GMS) (Groundwater Modeling Team) that does a Galerkinn formulation for flow and an Eulerian-Lagrangian approach for transport. Various boundary conditions are allowed, as well as density dependent coupled flow and transport for modeling salt-water intrusion. Legacy programs are especially difficult to parallelize effectively as they tend to be less structured, and almost all the algorithms are inherently serial. This paper describes some of the techniques used to produce a completely parallelized version of FEMWATER using the Message Passing Interface (MPI).

2. Flow equations. Pressure head in FEMWATER is modeled by applying conservation of mass to obtain,

$$\frac{\rho}{\rho_0} F \frac{\partial h}{\partial t} = \nabla \cdot \left[\boldsymbol{K} \cdot \left(\nabla h + \frac{\rho}{\rho_0} \nabla z \right) \right] + \sum_{m=1}^{fss} \frac{\rho_m^*}{\rho_0} Q_m \delta(\boldsymbol{r} - \boldsymbol{r}_m) \tag{2.1}$$

$$h = \frac{p}{\rho_0 g} \tag{2.2}$$

* U. S. Army Engineering Research and Development Center, Information Technology Laboratory, Vicksburg, MS 39180.
tracyf@wes.army.mil

(2.3) $$F = n\frac{dS}{dh} + S\alpha' + \theta\beta'$$

(2.4) $$\alpha' = \rho_0 g\alpha$$

(2.5) $$\beta' = \rho_0 g\beta$$

where

α = compressibility of the soil medium.

β = compressibility of water.

δ = Dirac delta function.

∇ = gradient operator.

fss = number of source / sinks for flow.

g = acceleration due to gravity.

h = pressure head.

h_i = pressure head at node i.

$\boldsymbol{K}$ = hydraulic conductivity tensor.

n = porosity.

N_i = interpolation function = 1 at node i and = 0 at all the other nodes.

p = pressure.

Q_m = quantity of flow at the mth source / sink node.

$\boldsymbol{r}$ = vector to (x, y, z) point in space.

$\boldsymbol{r}_m$ = location of the mth source / sink node.

ρ = density with contaminant.

ρ_0 = density without contaminant.

ρ_m^* = density of mth source / sink.

S = saturation.

t = time.

θ = moisture content.

What makes Eq. 2.1 so challenging to solve is that $\boldsymbol{K}$ and θ are functions of h, making the equation highly nonlinear.

Starting with the usual finite element approximation for pressure head

$$h = \sum_{i=1}^{N} N_i h_i \tag{2.6}$$

where N is the number of node points and N_i is an interpolation function with $N_i = 1$ at node i and $N_i = 0$ at all the other nodes. The Galerkin finite element equation is obtained by multiplying Eq. 2.1 by N_i, integrating over the volume of the mesh V, and then integrating by parts. The equation for the ith node becomes

$$\begin{aligned} &\sum_j \left(\int_V N_i \frac{\rho}{\rho_0} F N_j \, dv \right) \frac{dh_j}{dt} + \sum_j \left[\int_V \nabla N_i \cdot \left(\boldsymbol{K} \cdot \nabla N_j \right) dv \right] h_j = \\ &\int_{S_2} N_i \hat{\boldsymbol{n}} \cdot \left[\boldsymbol{K} \cdot \left(\nabla h + \frac{\rho}{\rho_0} \nabla z \right) \right] ds - \int_V \nabla N_i \cdot \left(\boldsymbol{K} \cdot \frac{\rho}{\rho_0} \nabla z \right) dv + \frac{\rho_i^*}{\rho_0} Q_i \end{aligned} \tag{2.7}$$

Here S_2 is the part of the surface where the flux is known. Specified pressure head occurs on S_1. Modifying Eq. 2.7 for the specified head nodes on S_1 and using a fully implicit representation for time, the matrix version of Eq. 2.7 becomes

$$[M]^{n+1}\left(\{h\}^{n+1} - \{h\}^n\right) + \Delta t [K]^{n+1} \{h\}^{n+1} = \Delta t \{Q'\}^n \tag{2.8}$$

where $[M]$ is analogous to a mass matrix in structures problems, $[K]$ is analogous to a stiffness matrix in structures problems, and $\{Q'\}$ is a collection of flow type terms for the right-hand side. Eq. 2.8 is the resulting system of equations, where both $[M]$ and $[K]$ are symmetric.

3. Parallel version for flow. The paradigm employed for creating the parallel version of FEMWATER is to modify the original source code as little as possible and have information exchanged across pe boundaries using MPI. The five most important parts to achieving the parallel version of FEMWATER are now described.

3.1 Partitioning. The finite element mesh must first be partitioned so that each processing element (pe) of the parallel computer has an equal share of the work. For a structured grid, this is relatively easy, as block partitioning can be done (see Figure 3.1.1). However, for an unstructured mesh, partitioning is considerably more difficult, so

the program METIS (Karypis) is used. A parallel program (part.c) was written to produce a file with a line for each node point stating which pe owns it.

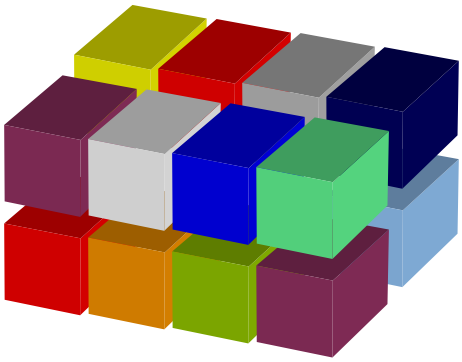

Figure 3.1.1. *Block partitioning.*

3.2. Ghost nodes. Border elements (see Figure 3.2.1) are those elements that have nodes that they own and nodes that they do not (ghost nodes). Each pe renumbers the owned nodes and elements to a local numbering system, and the ghost nodes and elements are placed at the end of their respective lists. The ghost nodes are also sorted so that all that belong to a particular pe are together. This allows the most efficient updating when the latest values of a certain variable such as pressure head are needed for the ghost nodes.

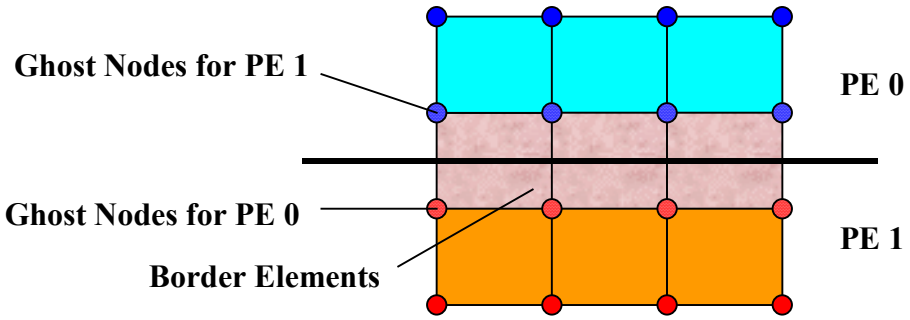

Figure 3.2.1. *Owned nodes, ghost nodes, and border elements.*

3.3. Boundary conditions. Boundary conditions can cause serious problems in performance as a specified head for a node in one pe can cause changes in equations for nodes in other pe's. However, if the border elements are kept by each pe rather than just one of the them, the boundary conditions are automatically properly done. A program

(prep.f) that reads in the original data describing the mesh and boundary conditions was written to produce a new set of data in the same format for each pe. This requires that local node and element numbers be put in the respective data files rather than the global ones so they remain consecutive. A file containing mappings of local, global, and ghost node and element numbers is also written for each pe. Thus when FEMWATER runs in parallel, each pe has its own data, and communications among pe's are required in only a few places.

3.4. Solvers. Solvers take a major part of the computational time, and therefore special effort must be taken in the parallel version for them. The original workhorse solver used in FEMWATER is a relaxation solver. As in this relaxation solver, most solvers need more iterations as the number of pe's increase, and so effective scaling is challenged. The relaxation solver can be parallelized by a domain composition type method where Dirichlet boundary conditions are applied to the ghost nodes, and each pe computes a new value for their owned nodes only. After each iteration cycle the ghost node values of pressure head are updated, and the process is continued until convergence.

This solver is generally slow converging, and it was found to be unacceptably slow for some problems such as those containing wells. Thus, two conjugate gradient (CG) solvers with a main diagonal (MD) pre-conditioner and an Incomplete Cholesky (IC) pre-conditioner (Dongara, Sorensen, and van der Vorst, 1998) were installed. A CG solver is easily parallelized by computing local matrix-vector products and doing global sums using MPI_ALLREDUCE at the appropriate times. The MD pre-conditioner creates no new parallel problems, but the IC pre-conditioner does. Given the system of equations to solve,

$$[A]\{x\} = \{b\} \tag{3.4.1}$$

First transform Eq. 3.4.1 as follows:

$$\begin{aligned} \{\tilde{x}\} &= [D]^{1/2}\{x\} \\ \{\tilde{b}\} &= [D]^{-1/2}\{x\} \\ [\tilde{A}] &= [D]^{-1/2}[A][D]^{-1/2} \end{aligned} \tag{3.4.2}$$

where

$[D]$ is the diagonal part only of $[A]$, and the non-zero elements of $[D]^{1/2}$ are

$$d_{ii}^{1/2} = \sqrt{d_{ii}} \tag{3.4.3}$$

This gives

$$[\tilde{A}]\{\tilde{x}\} = \{\tilde{b}\} \tag{3.4.4}$$

The pre-conditioner becomes

$$[\tilde{K}] = ([I] + [\tilde{L}])([I] + [\tilde{U}]) \tag{3.4.5}$$

where [*I*] is the identity matrix, $[\tilde{L}]$ is the lower triangular part of $[\tilde{A}]$, and $[\tilde{U}]$ is the upper triangular part of $[\tilde{A}]$. Using this pre-conditioner requires the solution of equations like

$$([I] + [\tilde{U}])\{y\} = \{r\} \tag{3.4.6}$$

and

$$([I] + [\tilde{L}])\{z\} = \{y\} \tag{3.4.7}$$

which certainly do not scale very well for sparse systems. What was done was to have each pe do its own sub-version of Eqs. 3.4.6 and 3.4.7 for its owned nodes only. This requires setting some terms in $[\tilde{L}]$ and $[\tilde{U}]$ to zero, which makes the parallel version of the IC pre-conditioner somewhere between the MD and full IC pre-conditioner. The resulting parallel solver has worked very well on all problems tested thus far.

3.5. Post-processing. The parallel FEMWATER program, fwpar.f, writes out individual files for each pe using their respective numbering systems. A program post.f was written to read in the output, assemble it into a global version, and write the same files that the serial version would write. As parallel I/O is a current research topic, later versions of fwpar.f may be better done by doing the parallel I/O directly.

4. Laboratory test problem. The flow part of FEMWATER was tested by a set of data from a laboratory test problem (Vauclin, Khanji, and Vachaud, 1979). The problem as shown in Figure 4.1 consists of flow in a homogeneous sand of saturated hydraulic conductivity K_s = 35 cm/hr. The relative hydraulic conductivity from experiment is given by

$$k_r = \frac{A}{A + (-h)^B}, \; A = 2.99(10^6), \; B = 5.00, \; h \le 0$$
$$k_r = 1, \; h \ge 0 \tag{4.1}$$

where the hydraulic conductivity tensor is now

$$\boldsymbol{K} = k_r K_s \boldsymbol{I} \tag{4.2}$$

Also,

$$\theta = n\frac{A}{A+(-h)^B},\ n=0.3,\ A=40.0,\ B=2.9,\ h\le 0$$
$$\theta = n,\ h\ge 0 \tag{4.3}$$

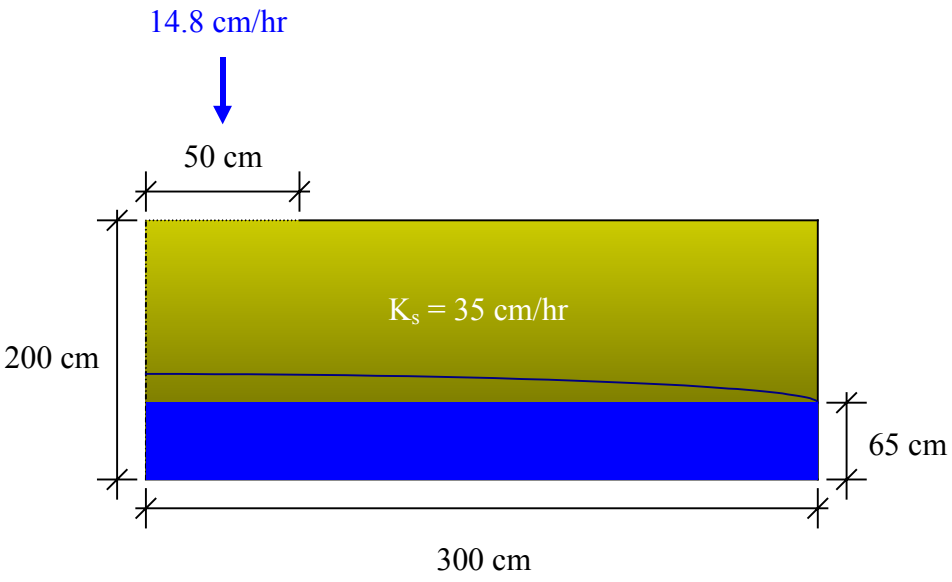

Figure 4.1. *Laboratory test problem.*

Due to symmetry, only half of the tank is shown. The tank is 65 m. tall and 600 m. long with a specified water level of 35 m. maintained at both left and right boundaries throughout the experiment. The top of the tank is initially covered, so the saturated water level is 35 m., and the soil is progressively drier as you go toward the top of the tank. Then the center 100 cm. of the tank is opened, and water at the rate of 14 cm/hr is released into the tank. The saturated water surface (free surface) will begin to rise as shown in Figure 4.1.

This problem barely converges with the relaxation solver, but the IC CG solver gives consistent convergence. B splines were implemented to represent the curves in Eqs. 4.2 and 4.3 over piecewise linear, and convergence was improved yet again. A Newton iteration scheme instead of the currently used Picarde method for the nonlinear iteration would most likely provide further improvement. Figure 4.2 shows a plot of the free surface as determined in the laboratory and as computed on a Cray T3E using the relaxation solver. Table 4.1 shows the scaled speed-up results where the problem size and the number of pe's are increased the same amount. Ideal is 100%. When keeping the problem size the same, going from a problem on 2 pe's to the same problem on 16 pe's on the Cray T3E, the parallel speed-up was 7.0. When going from a problem on 4 pe's to the same problem on 32 pe's on the Cray T3E, the parallel speed-up was 6.8.

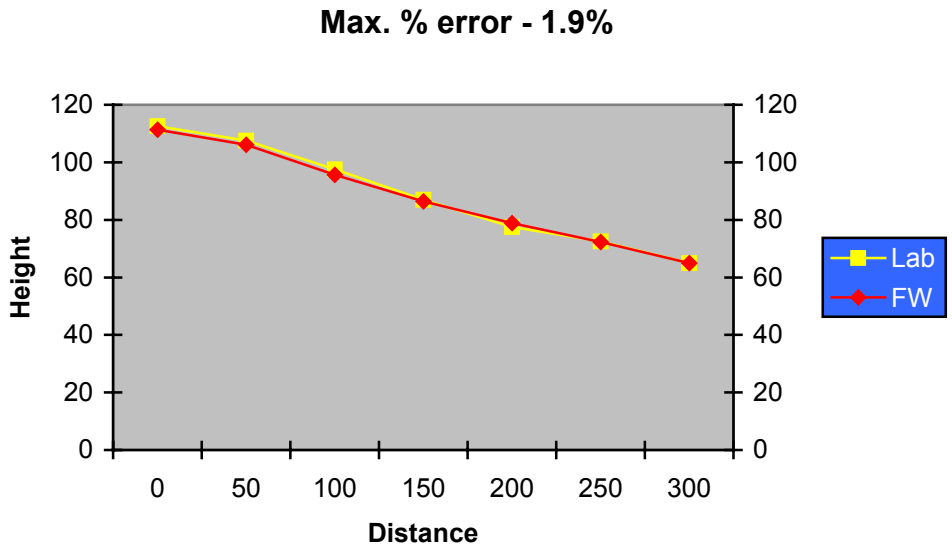

Figure 4.2. *Free surface after 4 hr.*

Number of pe's	Scaled speed-up (%)
1	100
2	86.2
4	81.5
16	74.6
32	67.6

Table 4.1. *Scaled speed-up.*

5. Transport equations. Transport in FEMWATER where a constant partition coefficient between the water and solid phase is modeled by

$$(\theta + \rho_b K_{ws})\frac{\partial C}{\partial t} + \boldsymbol{V}\cdot\nabla C + \nabla\cdot(\theta \boldsymbol{D}\cdot\nabla C) + (\theta C + \rho_b K_{ws})\left(\alpha'\frac{\partial h}{\partial t} + \lambda\right)C +$$

(5.1)
$$(\theta K_w + \rho_b k_s K_{ws})C + \left[\sum_{m=1}^{fss}\frac{\rho_m^*}{\rho}Q_m\delta(\boldsymbol{r}-\boldsymbol{r}_m) - \frac{\rho_0}{\rho}\nabla\left(\frac{\rho}{\rho_0}\right)\cdot\boldsymbol{V} - F\frac{\partial h}{\partial t}\right]C +$$
$$\frac{\partial\theta}{\partial t}C = \sum_{m=1}^{tss}C_m^* Q_m\delta(\boldsymbol{r}-\boldsymbol{r}_m)$$

(5.2)
$$\theta\boldsymbol{D} = a_T|\boldsymbol{V}|\boldsymbol{\delta} + \frac{(a_L - a_T)}{|\boldsymbol{V}|}\boldsymbol{V}\boldsymbol{V} + a_m\tau\boldsymbol{\delta}$$

$$\boldsymbol{V} = -\boldsymbol{K} \cdot \left(\frac{\rho_0}{\rho} \nabla h + \nabla z \right) \tag{5.3}$$

where

a_L = longitudinal diffusivity.

a_m = molecular diffusion coefficient.

a_T = transverse diffusivity.

C = concentration.

C_m^* = concentration of the contaminant in the water at the m source / sink..

$\boldsymbol{\delta}$ = Kronocker Delta tensor.

K_s = bidegridation rate for the solid phase.

K_w = bidegridation rate for the liquid phase.

K_{ws} = partition coefficient between the liquid and solid phases.

λ = decay constant.

ρ_b = bulk density.

τ = tortuosity.

$\boldsymbol{V}$ = discharge velocity or flux.

Eq. 5.1 is solved numerically by first doing a Lagrangian step and then an Eulerian one. This is done by operator splitting as follows:

$$\begin{aligned} &(\theta + \rho_b K_{ws}) \frac{C^{n+1} - C^*}{\Delta t} + (\theta + \rho_b K_{ws}) \frac{\partial C^*}{\partial t} + \boldsymbol{V} \cdot \nabla C^* + \\ &\nabla \cdot \left(\theta \boldsymbol{D} \cdot \nabla C^{n+1} \right) + T C^{n+1} \approx \sum_{m=1}^{tss} C_m^* Q_m \delta(\boldsymbol{r} - \boldsymbol{r}_m) \end{aligned} \tag{5.4}$$

where

T = the collection of terms multiplying C.

C^* = concentration after the Lagrangian step.

C^n = concentration at time t = nΔt.

C^{n+1} = concentration at time t = (n+1)Δt.

The Lagrangian step is done by solving

$$(5.5) \qquad (\theta + \rho_b K_{ws})\frac{\partial C^*}{\partial t} + \boldsymbol{V} \cdot \nabla C^* = 0$$

directly, which is back-tracking along the flow path. That is,

$$(5.6) \qquad C^*(\boldsymbol{r}, t) = C\left(\boldsymbol{r} - \frac{1}{\theta + \rho_b K_{ws}}\boldsymbol{V}, t\right)$$

This leaves the following Eulerian step to solve.

$$(5.7) \qquad (\theta + \rho_b K_{ws})\frac{C^{n+1} - C^*}{\Delta t} + \nabla \cdot \left(\theta \boldsymbol{D} \cdot \nabla C^{n+1}\right) + TC^{n+1} = \sum_{m=1}^{tss} C_m^* Q_m \delta(\boldsymbol{r} - \boldsymbol{r}_m)$$

6. Parallel version for transport. Eq. 5.7 is the same type as Eq. 2.1, so this poses no new complication. However, the backtracking step of Eq. 5.6 does. This is because the flow path could lead the computations across one or more pe boundaries. As backtracking does not involve a system of equations to solve, it is rather fast. The solution is to check in only the elements that contain the given node where C^* is to be computed. If the flow line has gone past these elements, reduce the backtracking time step until all backtracking falls inside these immediately neighboring elements, which are present in memory in each pe because of the overlapping already needed for flow and transport boundary conditions. Repeat these sub-steps until the full Δt is completed. This has worked very well for all problems tested thus far. Future papers will present detailed results.

REFERENCES

[1] J. J. DONGARA, D. C. SORENSEN, AND H. A. VAN DER VORST, *Numerical Linear Algebra for High-Performance Computers*, SIAM, Philadelphia, pp. 203-204, 1998.

[2] GROUNDWATER MODELING TEAM, *GMS*, http://chl.wes.army.mil/software/gms/, Engineering Research and Development Center (ERDC), MS.

[3] G. KARYPIS, *METIS*, http://www-users.cs.umn.edu/~karypis/metis/metis.html, University of Minnesota, MN.

[4] H. J. LIN , D. R. RICHARDS, C. A. TALBOT, G. T. YEH, J. R. CHENG, H. P. CHENG, AND N. L. JONES, *FEMWATER: A Three-Dimensional Finite Element Computer Model for Simulating Density-Dependent Flow and Transport in Variably Saturated Media*, Technical Report CHL-97-12, Engineering Research and Development Center (ERDC), MS, July 1997.

[5] M. VAUCLIN, D. KHANJI, AND G. VACHAUD, *Experimental and Numerical Study of a Transient Two-Dimensional Unsaturated-Saturated Water Table Recharge Problem*, Water Resources Research, 15(5), pp. 1089-1101, 1979.